

继承

- ◆ 为什么使用继承
- ◆ 如何使用继承
- ◆ 理解继承
- ◆ 继承的单根性和传递性
- ◆ 在何处使用继承
- ◆ 方法重写
- ◆ 继承中的构造方法
- ◆ 抽象类
- ◆ 抽象方法

为什么使用继承

▶ 这两个类图有什么问题？

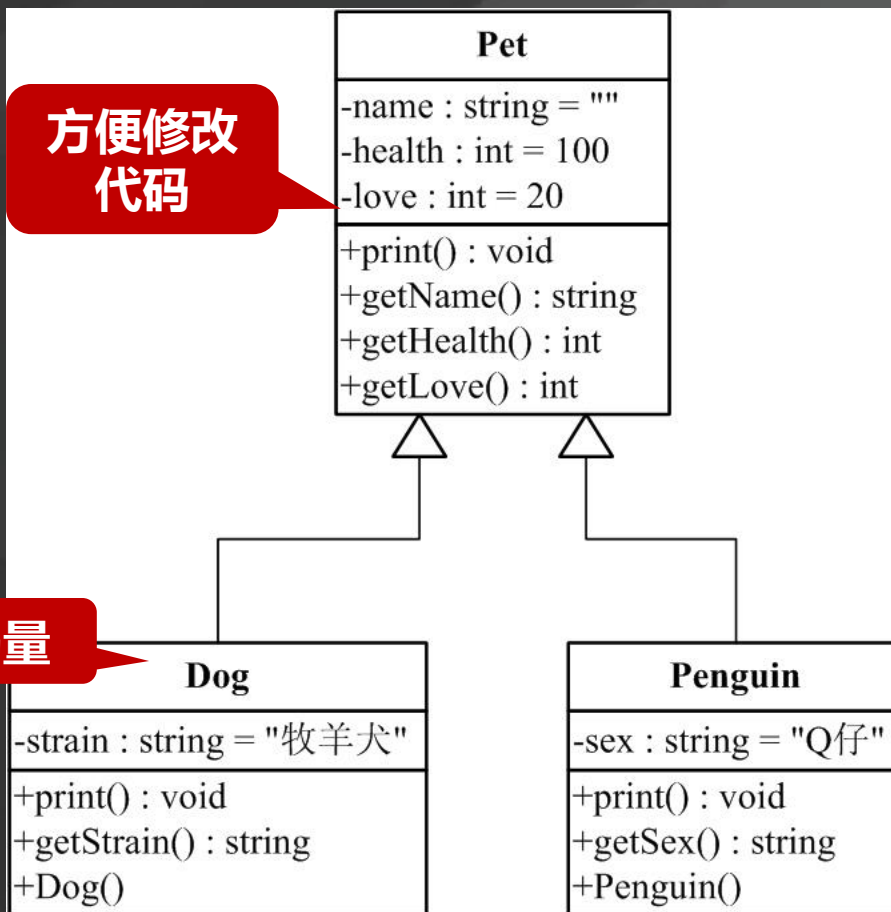
Dog	Penguin
<ul style="list-style-type: none">- name:String- health:int- love:int- strain:String	<ul style="list-style-type: none">- name:String- health:int- love:int- sex:String
<ul style="list-style-type: none">+ print():void+ getName():String+ getHealth():int+ getLove():int+ getStrain:String+ Dog()	<ul style="list-style-type: none">+ print():void+ getName():String+ getHealth():int+ getLove():int+ getSex():String+ Penguin()

将重复代码
抽取到父类
中

使用继承优化设计

为什么使用继承

使用继承优化后：



子类与父类是is-a关系

◆ 如何使用继承

▶ 使用继承

▶ 编写父类

```
class Pet {  
    //公共的属性和方法  
}
```

▶ 编写子类，继承父类

```
class Dog extends Pet {  
    //子类特有的属性和方法  
}
```

只能继承一个父类

```
class Penguin extends Pet {  
}
```

继承关键字

▶ 子类访问父类成员

使用`super`关键字

▶ 访问父类属性

```
super.name;
```

`super`代表
父类对象

▶ 访问父类方法

```
super.print();
```

▶ 如何在子类中调用父类的构造方法？

可以被默
认添加

```
super();
```

```
super(参数表);
```

只能是构造方法
的第一条语句

▶ 有些父类成员不能继承

- ▶ private成员

- ▶ 子类与父类不在同包，使用默认访问权限的成员

- ▶ 构造方法

▶ 访问修饰符protected

- ▶ 可以修饰属性和方法
- ▶ 本类、同包、子类可以访问

▶ 访问修饰符总结

访问修饰符	本类	同包	子类	其他
private	√			
默认(friendly)	√	√		
protected	√	√	√	
public	√	√	√	√

► 继承后的初始化顺序



► 阅读代码，说出运行结果

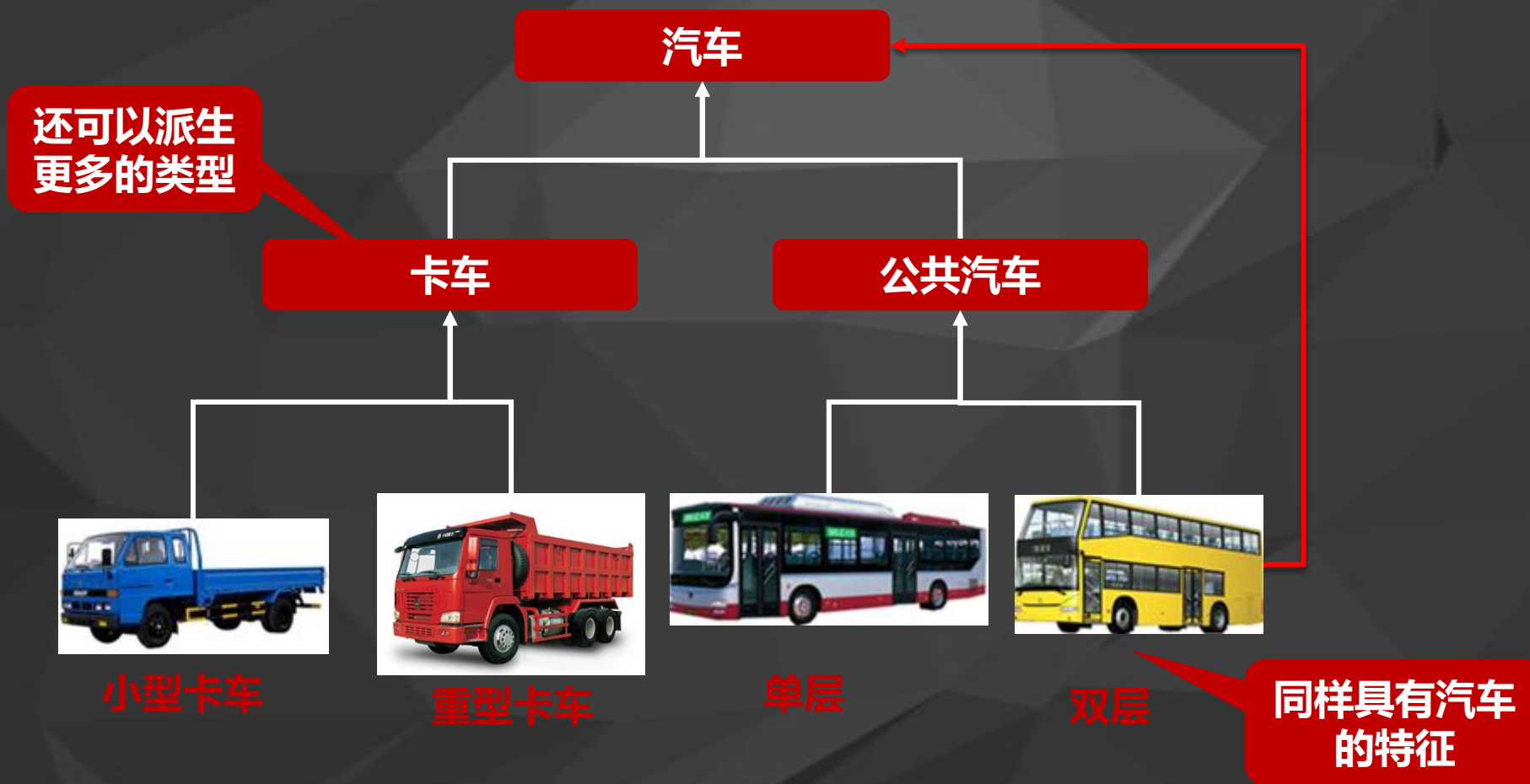
```
class Car {  
    private int site = 4; //座位数  
    Car(){  
        System.out.println ("载客量是"+site+"人");  
    }  
    public void setSite(int site){  
        this.site = site;  
    }  
    void print(){  
        System.out.print("载客量是"+site+"人");  
    }  
}
```

载客量是4人
载客量是20人

```
class Bus extends Car {  
    Bus(int site){  
        setSite(site);  
    }  
}
```

```
public static void main(String[] args) {  
    Bus bus = new Bus(20);  
    bus.print();  
}
```

继承的单根性和传递性



◆ 在何处使用继承

- ▶ 模拟现实世界的关系
- ▶ 便于重用和扩展已彻底测试的代码，且无需修改
- ▶ 结构更清晰

继承是代码重用的一种方式

将子类都有的属性和行为放到父类中

▶ 子类中有和父类相同签名的方法，会如何？

子类**重写**父类方法

◆ 继承中的构造方法

▶ 构造方法也会被重写吗？

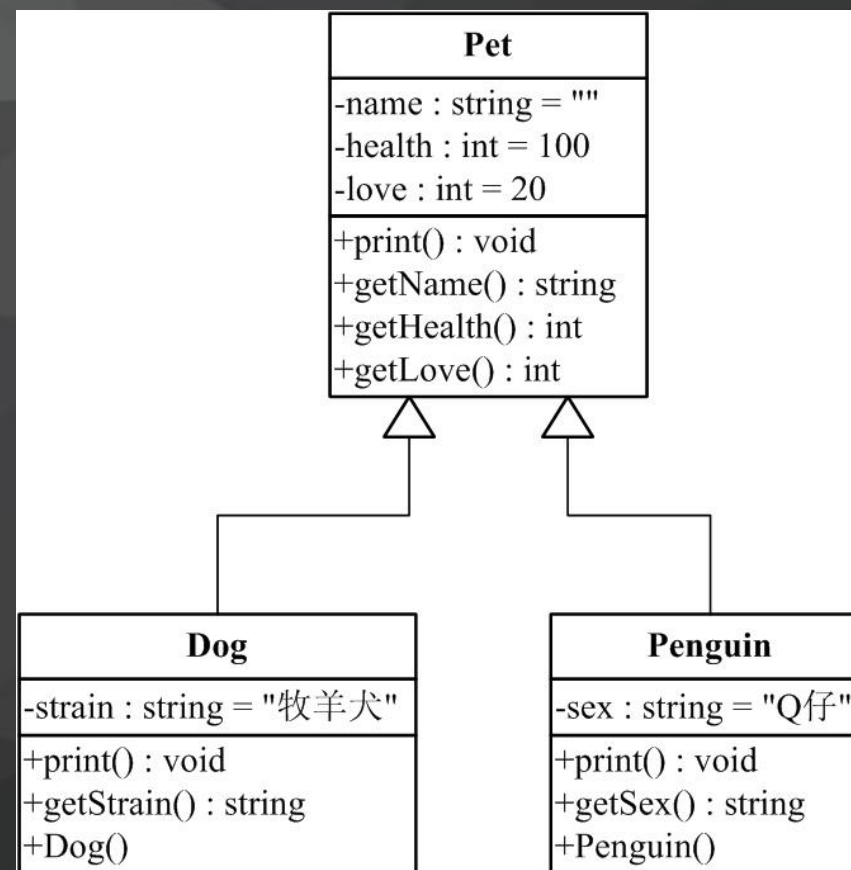
不能被继承因此不能重写

▶ 训练要点：

- ▶ 继承
- ▶ 子类重写父类方法
- ▶ 理解继承中的初始化过程

▶ 需求说明：

- ▶ 使用继承实现Dog类和Penguin类
- ▶ 打印宠物信息



▶ 以下代码有什么问题？

```
Pet pet = new Pet ("贝贝",20,40);  
pet.print();
```

实例化Pet没有
意义

▶ Java中使用抽象类，限制实例化

```
public abstract class Pet {  
}
```


▶ 以下代码有什么问题？

```
public abstract class Pet {  
    public void print() {  
        //...  
    }  
}
```

每个子类的
实现不同

▶ **abstract**也可用于方法——抽象方法

- ▶ 抽象方法没有方法体
- ▶ 抽象方法必须在抽象类里
- ▶ 抽象方法必须在子类中被实现，除非子类是抽象类

```
public abstract void print();
```

没有方法体

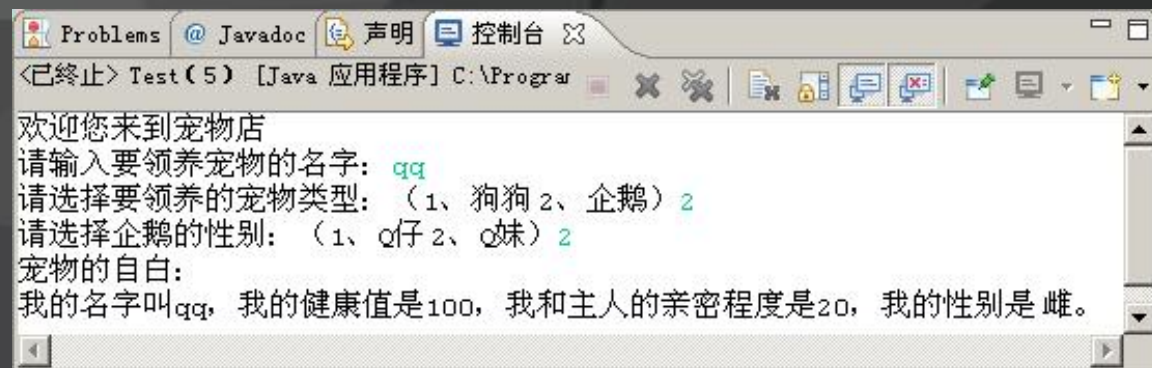
◆ 指导——抽象Pet类

▶ 需求说明：

- ▶ 修改Pet类为抽象类
- ▶ 修改Pet类的print()方法为抽象方法
- ▶ 输出Dog信息

▶ 实现思路

- ▶ 修改Pet类为抽象类，修改print()为抽象方法
- ▶ Dog类继承Pet类，实现print()方法
- ▶ 运行测试
- ▶ 注释掉Dog类中print()方法，运行测试类查看错误信息
- ▶ 编写注释

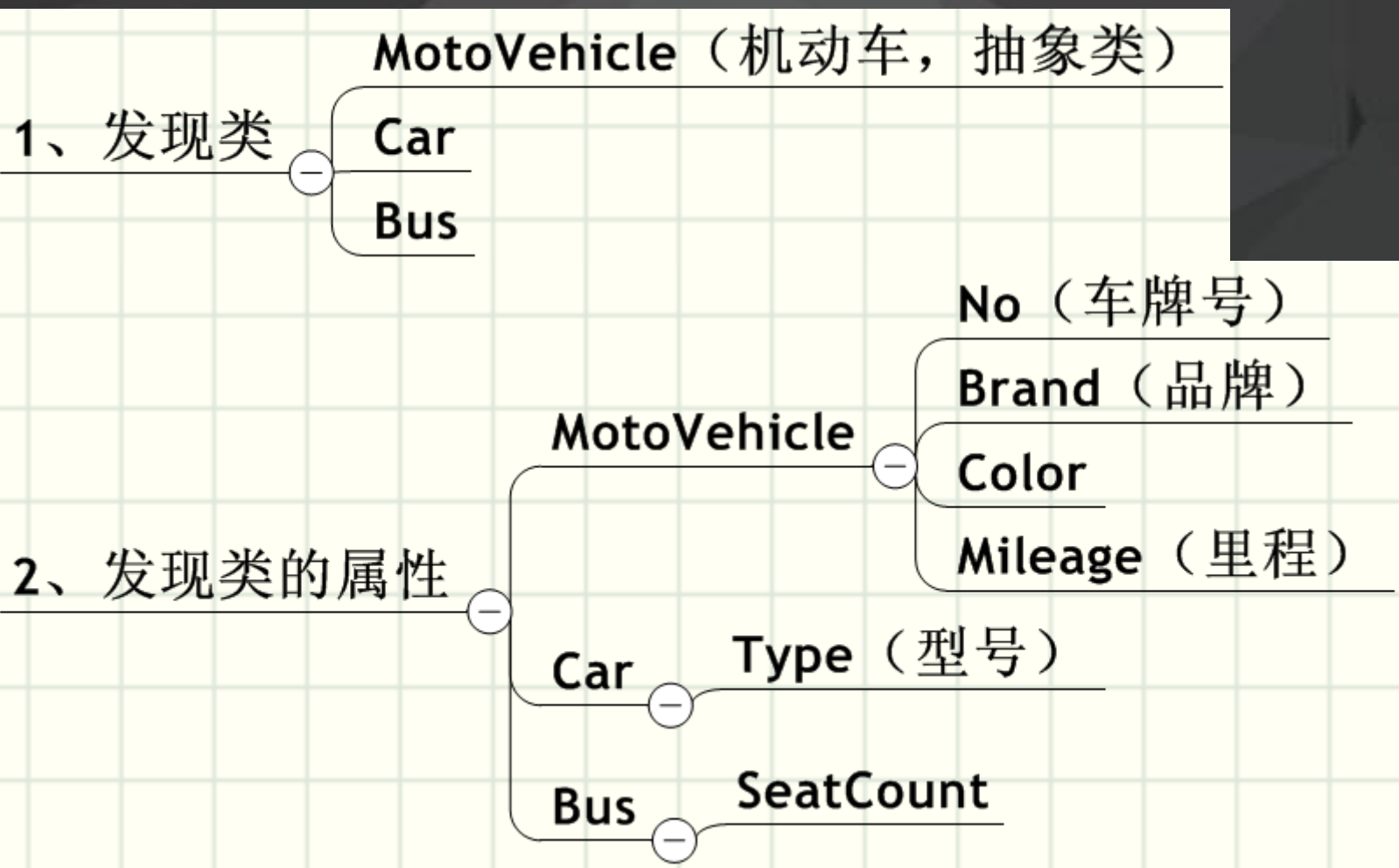


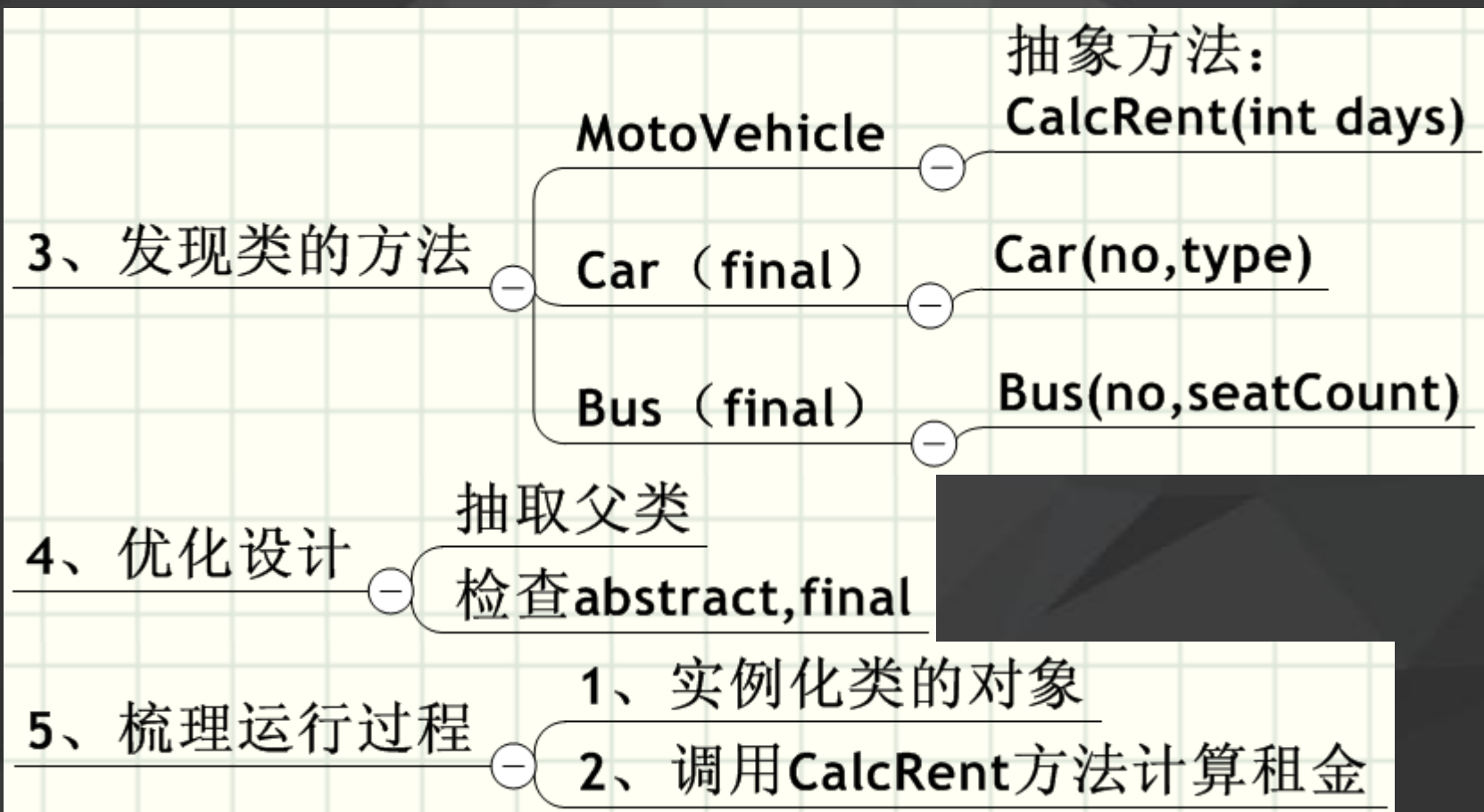
```
Problems  @ Javadoc  声明  控制台  已终止> Test (5) [Java 应用程序] C:\Program
欢迎您来到宠物店
请输入要领养宠物的名字: qq
请选择要领养的宠物类型: (1、狗狗 2、企鹅) 2
请选择企鹅的性别: (1、q仔 2、q妹) 2
宠物的自白:
我的名字叫qq, 我的健康值是100, 我和主人的亲密程度是20, 我的性别是雌。
```

- 某汽车租赁公司出租多种车辆，车型及租金情况如下：

	轿车			客车（金杯、金龙）	
车型	别克商务舱 GL8	宝马550i	别克林荫大道	≤16座	>16座
日租费 (元/天)	600	500	300	800	1500

- 编写程序实现计算租赁价

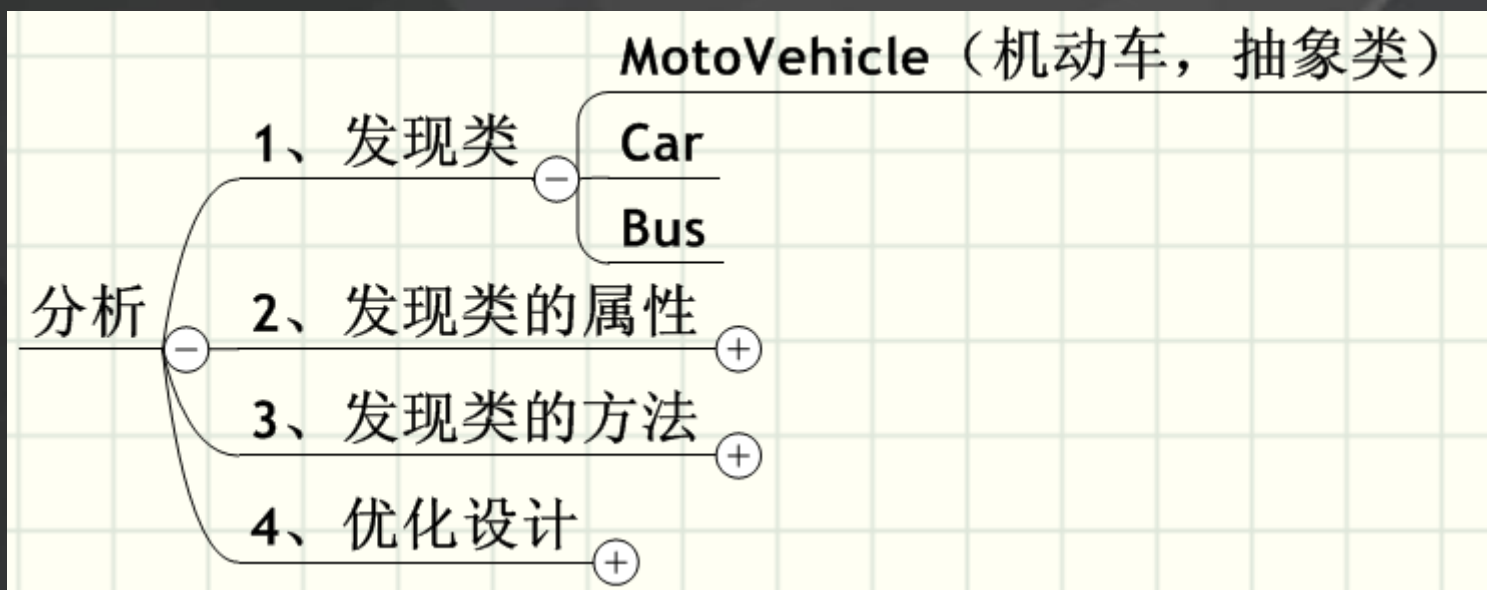




练习一编写类

需求说明：

根据分析编写MotoVehicle、Car、Bus类



◀ 练习——编写测试代码运行

- ▶ 需求说明：
- ▶ 编写测试代码运行

5、梳理运行过程

1、实例化类的对象

2、调用CalcRent方法计算租金

- ▶ 如何继承一个类？
- ▶ 继承有什么好处？
- ▶ 抽象类和抽象方法的特点是什么？
- ▶ 面向对象设计的步骤是什么？

谢 谢

